Christoph Döpmann, Sebastian Rust, Florian Tschorsch

Exploring Deployment Strategies for the Tor Network [Extended Version]

First published as

Döpmann, Christoph; Rust, Sebastian; Tschorsch, Florian (2018): Exploring Deployment Strategies for the Tor Network [Extended Version]. Cryptology ePrint Archive 2018/661. https://ia.cr/2018/661.

This is an extended version of the following publication:
Christoph Döpmann; Sebastian Rust; Florian Tschorsch (2018). Exploring Deployment Strategies for the Tor Network. 2018 IEEE 43rd Conference on Local Computer Networks (LCN). https://doi.org/10.1109/lcn.2018.8638043.

Exploring Deployment Strategies for the Tor Network [Extended Version]

Christoph Döpmann
Distributed Security Infrastructures
Technische Universität Berlin
christoph.doepmann@campus.tu-berlin.de

Sebastian Rust
Institut für Informatik
Humboldt-Universität zu Berlin
rustseba@informatik.hu-berlin.de

Florian Tschorsch
Distributed Security Infrastructures
Technische Universität Berlin
florian.tschorsch@tu-berlin.de

***Abstract*—In response to upcoming performance and security challenges of anonymity networks like Tor, it will be of crucial importance to be able to develop and deploy performance improvements and state-of-the-art countermeasures. In this paper, we therefore explore different deployment strategies and review their applicability to the Tor network. In particular, we consider flag day, dual stack, translation, and tunneling strategies and discuss their impact on the network, as well as common risks associated with each of them. In a simulation based evaluation, which stems on historical data of Tor, we show that they can practically be applied to realize significant protocol changes in Tor. However, our results also indicate that during the transitional phase a certain degradation of anonymity is unavoidable with current viable deployment strategies.**

***Keywords*—Internet security, overlay networks**

I. Introduction

The importance of anonymous communication networks (*ACN*s) such as Tor [1] has steadily grown over the past years. In order to keep up with current and future requirements—for example strengthening anonymity against increasingly powerful adversaries, or improving performance to broaden the user base—it is of decisive importance to carry out active research in this field. Great technological advances will fail to reach practical importance, though, if they cannot actually be deployed in a safe and practical manner. Yet, this is an aspect that is oftentimes neglected in research.

With ACNs in general, deployment of system changes becomes an especially difficult task due to two inherent system properties: Firstly, special care must be taken not to exclude any users from the system. Secondly, it constitutes a non-trivial technical problem that consists in maintaining the network's functionality without breaking anonymity during the deployment process. The latter is hard because, like Tor, ACNs are oftentimes highly decentralized systems, in which no single entity has control over individual entities. Incremental upgrades might easily result in a network split or otherwise harm the anonymity set that is essential to protect its users.

The main focus of this paper lies on the deployment of changes that affect network communication or the infrastructure within Tor. While changes to other parts of Tor are equally important, these are generally easier to deploy. For instance, cryptographic primitives used in Tor have been successfully replaced [2]. Tor's ability to differentiate between types of cells made this transition relatively straightforward to implement. On the other hand, as soon as the network layer is touched by a protocol change, this can have tremendous, inherently backwards-incompatible consequences. Examples of such changes that may be desirable to adopt in the future include fundamental changes to either connection handling [3], congestion control [4], or even substituting the transport protocol [5], [6].

In this paper, we first investigate how *flag day transitions* can be applied in the context of Tor, yielding a potential way of introducing arbitrary protocol changes. Pointing at the deficiencies associated with this approach, we also consider coexistence-based approaches like *dual stack*, *translation* and *tunneling*, analyzing their strengths and drawbacks and relating them to real proposed network changes. We validate their applicability by carrying out a simulation-based evaluation. Our work indicates that certain network changes possibly cannot be deployed without accepting some degradation of anonymity during the transitional phase.

The contributions of our paper can be summarized as follows. We give an overview of deployment challenges in Tor and identify existing mechanisms that can be leveraged to deploy small, compatible changes. We investigate strategies suitable for deploying more fundamental changes to the Tor protocol and juxtapose their advantages and individual risks regarding security and anonymity. Furthermore, we evaluate the proposed approaches' impact on the network and emphasize the importance of carefully choosing an appropriate deployment strategy during development.

The remainder of this work is structured as follows. In Section II, we review related work. In Section III, we describe the Tor ecosystem and provide first starting points for deployments. Section IV deals with various deployment strategies and discusses a suite of general approaches towards the deployment in Tor, before evaluating their impact on the network in Section V. We conclude our work in Section VI.

II. Related Work

The deployment of software enhancements has attracted considerable amount of attention, both within the industry and scientific research. For instance, [7] presents a framework for categorizing deployment strategies. Over the past years, the process of software deployment has been revolutionized

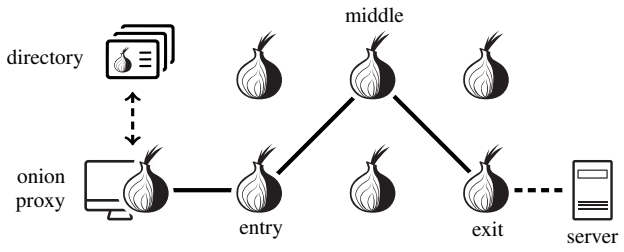

Fig. 1. The Tor network. Each onion symbol represents a relay. The list of all relays is provided by the directory. Clients typically select three relays to construct a circuit. Onion routing ensures anonymity of the transmitted data.

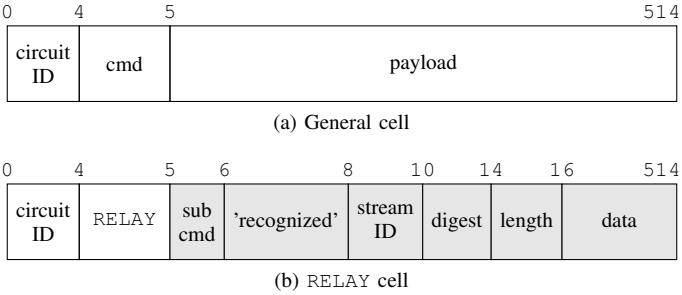

Fig. 2. Structure of Tor cells. (a) depicts the general cell structure, while (b) depicts a so-called RELAY cell. The grey part is onion encrypted.

first by the introduction of virtualization [8], and later by containerization techniques [9].

While there has been less attention on the deployment of network-based software, especially evolving network protocols, this field has not been ignored, either. As a prime example, the still ongoing transition from IPv4 to IPv6 has been generally discussed in [10] and motivated several suitable approaches [11]. To some extent, both contributions inspired the strategies proposed in this work. We however focus on anonymous communication networks in general and the Tor network in particular, which induce additional requirements regarding security and anonymity.

In the context of Tor, [12] constitutes a concise survey of proposed extensions to the Tor protocol. It provides a useful overview on what kinds of changes we have to take into account when constructing a deployment scheme for Tor.

The challenge of deploying changes to the Tor network has barely been covered by previous research, though. [13] reports on challenges of deploying a low-latency anonymity network, but mostly deals with general challenges, including the social challenges, and not specifically to changing an *existing* system. To the best of our knowledge, [14] is the only work that covers how a specific incremental update to the Tor software was realized. In contrast, our work has a strong focus on the technical point of view of deployment strategies and strives to explore generic solutions to realize complex protocol evolvement in Tor.

III. The Tor Ecosystem

We give a concise overview of the Tor ecosystem, including a brief description of components and their interaction. This section also provides first hints on integrating new features.

A. Onion Routing

Tor is based on the concept of *onion routing* [15], which in turn was strongly inspired by the principle of *mix cascades* [16], introduced by David Chaum in 1981.

The main idea consists in building a multi-hop tunnel that is used to carry the payload over a series of intermediate *onion routers* (OR), casually referred to as relays. In Tor, such a cryptographically secured tunnel, which is called a *circuit*, typically consists of three relays as illustrated in Figure 1. A circuit is constructed by extending each hop incrementally, like a telescope, where each relay removes or adds one layer of encryption. The onion routing protocol and the circuit construction ensures that each relay in a circuit only knows its immediate predecessor and successor, which eventually provides anonymity.

For bootstrapping communication as well as for authenticating relays, Tor relies on a directory service, which is maintained by a small number of trusted authorities.

B. Data Transport

Tor employs the principle of circuit switched networking to realize the data transport. Accordingly, so-called *cells* are passed along a virtual circuit and are used to carry payload as well as control signals. Between relays, they are exchanged over TLS-secured TCP connections.

For anonymity reasons, cells have a fixed size (originally 512 bytes and 514 bytes in the most recent version) and are padded with null bytes if necessary.[1] The general cell structure, shown in Figure 2a, consists of three parts: The *circuit ID* is a per-hop circuit identifier, enabling relays to match cells to circuits. The *command* denotes the cell type, defining how the cell should be handled. The *payload* contains command-specific data, e.g. parameters. Tor defines numerous different cell types (or commands), such as `CREATE(2)` for creating a new circuit or adding a new relay to it, or `RELAY` to carry encapsulated data or commands over a circuit.

`RELAY` cells are particularly interesting because their payload is "onion encrypted", which means that only the end points of a circuit are able to decipher it. They are therefore used for tunneling the client's data through the network, but also for implementing signaling that acts on an end-to-end basis. For instance, Tor implements congestion control and circuit extension requests by using `RELAY` cells. In order to support the different operations, the payload of `RELAY` cells contains another header and command identifier as depicted in Figure 2b.

C. Protocol Versions in Tor

The Tor protocol [17] can be regarded as a suite of different subprotocols that describe different aspects of the Tor network. Table I lists the subprotocols Tor currently comprises.

[1] As an exception, Tor also defines *variable-length cells* for certain purposes, which are not required to be padded to the fixed length. The `VERSIONS` cell type is an example for this.

TABLE I
SUBPROTOCOLS DEFINED IN THE TOR PROTOCOL SPECIFICATION.

Subprotocol	Versions	Scope
`Link`	1-5	Connection handling between relays
`LinkAuth`	1-3	Cryptographic scheme used for authenticating relays to each other
`Relay`	1-2	Circuit handling (creation, data relaying, ...)
`HSIntro`	3-4	Hidden services introduction points
`HSRend`	1-2	Hidden services rendezvous points
`HSDir`	1-2	Hidden services description documents
`DirCache`	1-2	Document fetching from directory caches
`Desc`	1-2	Descriptor documents as available from directories
`Microdesc`	1-2	Microdescriptor documents
`Cons`	1-2	Consensus documents

Among these, the `Link` and `Relay` protocols may be regarded as the ones most prone to future changes that affect the core networking functionality of the Tor network. This is because they define the way data is carried through the network, directly impacting on the reachable performance and anonymity properties. Thus, they will be especially subject to improvements amended by ongoing and future research. Note that the protocols defining documents used within Tor (`Desc`, `Microdesc` and `Cons`) are constructed in a way that allows for new data items to be added in a backwards-compatible way and only need to change in the event of grave syntactic or semantic changes of the documents themselves.

Tor provides basic mechanisms to support evolution of these subprotocols. The single protocol versions that are supported by each relay are published in the directory data, together with a span of minimum recommended and minimum required supported version. However, Tor relays do not generally rely on this data for normal operation. Instead, the protocol versions are engineered in a way that enables seamless interaction, agreeing on a common version when necessary. For example, the `Link` protocol version is at the time of writing chosen by exchanging `VERSIONS` cells, but was previously determined through subtle, compatible differences in the handshake.

In general, Tor offers two ways of rolling out protocol changes in a backwards-compatible way. Firstly, new cell types and `RELAY` cell subcommands can easily be introduced. Unknown cell types will be ignored by legacy nodes. Secondly, support for specific protocol versions can be signaled through the relay descriptors in the directory.

D. Roles and Entities

In order to reveal the challenges that arise when designing an appropriate deployment scheme for the Tor network, it is important to understand that Tor is an inherently decentralized system. As such, various different entities take part in the network. A deployment strategy must take this into account, ensuring that the rollout process can handle differently-paced adaptation throughout the network.

Onion Routers: Onion routers (or relays) form the backbone of the Tor network. They implement the onion routing protocol and pass traffic through an overlay network. Relays are typically run by volunteers, implying that their resources and level of maintenance differ significantly. In Tor, relays can take different roles. *Non-exit relays* purely act within the network, carrying data between other relays. In contrast, *exit relays* are the circuits' endpoint, facing the Internet. The heterogeneity of relays and the different roles make deployments challenging.

Clients: The client software is the user's entry point into the Tor network, oftentimes referred to as the *Onion Proxy*. Like the exit relay, the client takes a special role as an endpoint of a circuit. Since the client builds the circuit, however, it knows all relays and is able to control the use of specific protocol versions or to coordinate an interoperability scheme.

Directories: Clients and relays rely on a directory service, which describes the current network state. In constant time intervals, currently once per hour, *directory servers* (also called *directory authorities*) publish a list of relays known to be active in the network. For each relay, so-called descriptors include various meta information that is published along with the relays' "contact details". Most importantly, the descriptors include public cryptographic keys that are used for authenticating relays. In addition, relays can be assigned *flags* that denote special roles within the network. Descriptors also list supported subprotocol versions for each relay.

The descriptors are assembled based on self-reported information by the relays. Directory servers collect these information and exchange their current view of the network with the other directories. The directory servers mutually agree on a common network state and publish the result as the *consensus*. The consensus is trusted by clients and relays if it is signed by more than half of the known directories. As of today, there is a fixed set of 10 (semi-)trusted directory servers, whose identities are distributed with the Tor software.

In order to distribute load, there are numerous *directory caches* that serve the resulting signed descriptor documents to clients and relays.

The Tor Project: The Tor Project is a non-profit organization that coordinates the development of Tor. While Tor is open source and a community-driven project, the role of The Tor Project in providing binary releases of the Tor software may not be underestimated. Through the distribution of software releases and updates, it comes closest to controlling the software that is run in the network. Note however, that they do not have ultimate control over the single nodes. After all, the software and protocols are open source, which is an integral part of the project's security strategy.

IV. DEPLOYMENT STRATEGIES

In this section, we develop general approaches towards deploying changes to the Tor network and relate them to existing Tor research, which serve as examples. Depending on the proposed change, different deployment challenges are to be tackled. In the context of Tor, there are two main problems that arise. Firstly, communication through Tor constitutes a *multi-hop* scenario, meaning that more than two nodes are actively involved into the data transfer. Far-reaching changes to the Tor protocol may require all relays on a circuit (or even the whole

network) to support the new feature. It is essential to have mechanisms in place for dealing with different versions in the network, stemming from incremental deployment. Otherwise, some pairs of relays will not be able to work together anymore. A situation like this is to be avoided under all circumstances as it would effectively result in a network split, heavily reducing the users' anonymity set. That said, maintaining the *anonymity set* is the second notable challenge. Tor is only effective if the users' communications are hidden by a sufficiently large set of relays and users, among which a single user's stream of communication is not identifiable. Therefore, backwards compatibility is of utmost importance.

There are two general approaches that we consider, flag day and co-existence.

A. Flag Day: Building a Global Switch

The *flag day* strategy consists in incrementally rolling out a software update but keeping it disabled until a coordinated, simultaneous activation. This way, any change to the network can be deployed, with no backwards compatibility required. Even changes as fundamental as deploying a new transport protocol would be feasible. While early versions of Tor carried out a flag day transition [18], developers did not actually implement a coordinated deployment process, but the update only resulted in backwards incompatibility. At today's network scale, however, a coordinated switch to new network behavior would be necessary.

The immediate and main challenge of flag day transitions consists in deciding when to carry out the activation. Hard-coding an activation time appears to be suboptimal as it does not give any guarantees on the share of the network that has adopted the change and can therefore continue to work after activation of the system change. Instead, the decision should be based on consensus or central activation, e.g. directed by the Tor directories. For example, as a condition for carrying out the switch, a minimum share of relays that have upgraded could be defined. The possibility of Sybil attacks to trigger a premature activation has to be taken into account, though.

While building a reliable flag day transition is far from trivial and generally undesired by the Tor community [19], it is conceivable and may have to be considered for large, breaking system changes. When designing a flag day transition, the goal is to minimize the impact on the network, which we will investigate in our evaluation.

Note, however, that this network perspective is not necessarily sufficient to evaluate the impact of carrying out a flag day. In particular, it has to be made sure that most *clients* support the novel behavior before switching over. This is a hard challenge because, in contrast to relays, the clients do not publish version information about themselves in the directory. With this major obstacle in mind, we realize that carrying out a flag day transition in a robust way, without putting an unknown number of end users to the risk of being excluded from the Tor network, is currently infeasible. This could, however, change in the future with the advent of *PrivCount* [20], which aims at enabling the privacy-preserving collection of user statistics.

B. Co-existence: Implementing Interoperability

In contrast to the previous all-or-nothing approach, strategies that are based on a co-existence model allow relays of different versions or feature sets to properly interoperate. This is a very desirable property since it mitigates the risk of splitting the network. In order to ease the future design and implementation of appropriate co-existence schemes, we present a categorization of different models.

While co-existence can be an integral part of a system change, e.g. the protocol change is inherently backwards compatible, we here focus on situations in which this is not a priori given. That is, we primarily focus on how the deployment strategy deals with circuits that comprise relays with different software versions. In the following, we call relays that have not (yet) upgraded *legacy* nodes.

Before diving into the discussion of different co-existence strategies, we would like to note another challenge: In order for one of the following strategies to handle circuits of both legacy and upgraded relays, interoperation between relays has to be achieved.

The most straightforward approach consists in using the publicly available directory information. Based on that, the client can coordinate interoperability during the construction of the circuit. Due to Tor's telescope-like circuit construction, clients can instruct each relay without breaking anonymity.

Another conceivable strategy could rely on hop-by-hop decisions. If possible, such a scheme may be preferable over coordination that relies on data from the directory. While the directory already contains a relay's set of supported subprotocol versions, being able to interoperate with local information only is considered to be more robust, as it is for instance agnostic about recent version changes not yet reflected in the directory. This point of view is also built into the current Tor protocol specification, for instance by agreeing on a common `Link` subprotocol by making use of `VERSIONS` cells. However, this way of locally deciding which interop scheme to use may not be possible for many far-reaching protocol changes.

In the following we assume that a suitable mechanism to negotiate an interoperability scheme is in place or can be built with reasonable effort.

Dual Stack: For the *dual stack* strategy, we assume that even nodes that have upgraded still support the legacy protocol and are able to use them selectively or in parallel. This way, a circuit can make use of the newest supported protocol version that all relays support. As a consequence, single circuits may already benefit from the newly deployed technology at a time when the rest of the network has not upgraded yet. Similarly, legacy nodes continue to work as before, resulting in the circuit to stick to the previous behavior until upgraded.

A dual stack approach can be used to deploy a wide range of complex changes to the network. However, it comes with certain drawbacks. Most prominently, the time it takes until clients as well as the network itself can benefit from the new feature, might be relatively long. We will quantify and evaluate this delay in our evaluation.

A dual stack strategy might also lead to weakened anonymity. For example, consider the case of an extensive change to the underlay transport protocol, e.g. the replacement of TCP with UDP. If this change is active for some circuits already while others still rely on TCP, an adversary may be able to exploit this. By observing which variant is used for a specific circuit, a passive network adversary or a malicious relay can deduce which relays may be involved in the circuit. When observing UDP instead of TCP, this would mean that only new, upgraded relays are part of the circuit. Such observations could potentially have a severe impact on the anonymity set that defines a user's degree of anonymity. While the UDP-vs-TCP example is an extreme case, more subtle changes like traffic pattern deviations may be exploited by a sophisticated adversary, as well.

We argue that this issue cannot completely be avoided (and also might apply to other deployment strategies). At least active relays will always be able to determine the protocol version in use—otherwise, a dual stack approach would not be necessary. Therefore, such relays could infer version information about distant nodes, impacting on the anonymity set. The degree to which this is problematic should be evaluated individually prior to introducing a dual stack system. Our evaluation methodology can be used as a starting point to quantify the impact.

Candidates for being deployed with a dual stack approach are for example *DefenestraTor* [4] and *BackTap* [6]. DefenestraTor solely aims at replacing Tor's end-to-end congestion control with a custom scheme based on hop-by-hop information. BackTap goes even further and introduces a novel transport protocol, which tries to tackle this and other performance issues in Tor. Both require the interaction of every relay on the circuit in order to work properly.

Translation: A different approach can be used when relays on a circuit have differing feature sets. That is, neither all nor none of the circuit's relays have upgraded yet. In this case, a *translation* between the protocol versions may be applied. Depending on the change that is to be deployed, this can provide a benefit of partially running the new protocol version, even when the whole circuit has not yet upgraded. For instance, assume the protocol enhancement works in a hop-by-hop manner and improves performance by applying it to this segment. With translation, clients can benefit from such an enhancement earlier.

Note that translation can be as simple as dropping or slightly modifying messages, but may just as well prove impossible to be done in a meaningful way. If compatibility can be achieved by converting between cell types, for example, the translation is straightforward to implement. On the other hand, if a protocol change requires end-to-end feedback, similar to Tor's congestion control, this could not necessarily be deployed using a translation scheme.

One real-world proposed Tor enhancement that could well be deployed relying on translation is *TCP-over-DTLS* [5]. It strives to improve latency in Tor by avoiding head-of-line blocking, which occurs in normal Tor operation due to the multiplexing of multiple circuits over single TCP connections. In order to achieve this, it relies on a user space TCP implementation whose emitted segments are then tunneled over *Datagram TLS (DTLS)*, which is based on UDP, to the next relay. Deploying this protocol change using a translation scheme is possible because the carried cells can simply be received via the novel DTLS stream, and forwarded to a legacy relay over a normal TCP connection. A circuit that consists of a mix of legacy and upgraded relays can therefore, to a certain degree, already benefit from TCP-over-DTLS without the need for the whole circuit to support it.

Tunneling: If the circuit consists of relays whose feature sets form a "gap", that is, a relay in the *middle* of the circuit still lacks support for the new feature, yet another strategy is conceivable. In this case, the protocol data that is exchanged between the novel relays around the gap may be *tunneled* through the circuit segment of one or more legacy nodes.

The core idea behind a scheme like this is the following: If the new feature relies on an end-to-end flow of information through the circuit, the mechanism may become usable even if not the whole circuit supports it, yet. If, for instance, the Tor client relies on feedback from the exit, this may be deployed even if not all relays in between have upgraded.

Tunneling is not necessarily a special case of (twofold) translation since it may be applicable when there are no two consecutive relays of the same version, in which case translation is impossible.

The ways tunneling can be implemented may vary greatly. If a new feature exclusively relies on the circuit endpoints communicating with each other, the implementation can be trivial. As Tor provides a mechanism for clients to transparently communicate with any relay within the circuit by using `RELAY` cells, this can be leveraged to bridge legacy relays. If such a solution is not applicable, it may instead be possible to encode the necessary information within other legacy cells.

Tunneling could, for instance, be applied to proposals like *UDP-OR* [21]. Like TCP-over-DTLS, UDP-OR proposes to use UDP as the underlying transport protocol. However, it is not meant to operate on a hop-by-hop basis, but end-to-end. More specifically, the UDP transport is used for carrying TCP segments from one end of the circuit to the other, that is, client and exit perform the TCP algorithm over the complete circuit. Intermediate relays are not actively involved in this behavior, and carry `RELAY` cells only. Therefore, UDP-OR could be tunneled through legacy relays by falling back to cell transport via TCP connections. The semantics of encapsulated data is only relevant to the endpoints of the circuit.

UDP-OR also constitutes an example showing that the approaches we present still require careful consideration when choosing them for a certain protocol change. In this specific case, tunneling UDP-OR cells over legacy TCP connections will eventually result in a TCP-over-TCP situation, whose performance could be worse than expected.

V. Evaluation

Having given an overview on several general approaches towards deploying changes to the Tor network, we now aim at quantifying the impact of each of them.

Note that the methods presented only constitute basic conceptual ideas. Implementing them for a specific deployment scenario will most probably require tailoring and fine-tuning. The applicability of each method and the significance of this evaluation may therefore greatly vary depending on the specific use case. Nevertheless, we aim at generating general insight on the benefits and drawbacks that are associated with the proposed strategies.

A. Methodology

In order to compare against a realistic model of relays' upgrade behavior, we base our evaluation on historical data of software version distribution within the Tor network. Thus, we make use of data collected by the *Tor Metrics* project [22]. We consider a time span of March 2013 to January 2018. Over this period of time, we track the share of relays that have upgraded to Tor version 0.2.5 or higher, which was released at the beginning of this time span. We chose to analyze the deployment of this particular Tor version for the following reasons: Firstly, the predecessor, version 0.2.4, had great prevalence within the network and is still run by a notable amount of onion routers, even years after its release. Deploying against such a persistent legacy system comes close to a realistic worst case scenario for deployment. Secondly, version 0.2.5 is the first version released under the current maintenance and release model of publishing versions with different long-term support every six months. Choosing the first of these versions gives us the maximum possible simulation duration while keeping the results comparable to today's situation.

For analyzing flag day transitions, we take a general evaluation approach that allows us to quantify a flag day's impact independently of a specific protocol change. We do so by leveraging generic Tor version distribution information.

The evaluation of co-existence schemes requires to operate on a more fine-grained, circuit-level scale. Since the effectiveness of these strategies is highly dependent on the configuration of every single circuit, we simulate the circuit selection. To this end, we use the *Tor Path Simulator* (TorPS), introduced in [23]. Given the historical Tor network data, it simulates the circuit selection process of clients and allows us to gather aggregate data on specific properties of the simulated circuits. In order to measure the applicability of each of the co-existence strategies for given circuit configurations, we assume prototypical protocol changes that may practically be deployed using one of the discussed approaches. We use the examples provided when introducing the different schemes as mental templates and derive a compatibility matrix, shown in Table II.

Throughout the evaluation of co-existence schemes, we assume the client software to be up to date. The study thus investigates to what degree a modern client can benefit from technological progress as it is rolled out in the Tor network over time.

TABLE II
Applicability of co-existence schemes.

Circuit configuration (entry-middle-exit)	**Applicable schemes** Dual Stack	Translation	Tunneling
○–○–○	✗	✗	✗
○–○–●	✗	✗	✓
○–●–○	✗	✗	✗
○–●–●	✗	✓	✓
●–○–○	✗	✓	✗
●–○–●	✗	✓	✓
●–●–○	✗	✓	✗
●–●–●	✓	✓	✓

● upgraded ○ legacy

B. Flag day

We first analyze the impact of a flag day strategy on the network. We define the following metrics: For any given point in time, we evaluate which network relays have upgraded. These are the relays that would keep working if a flag day was carried out at that time. From this set, we calculate the *share of relays* that are retained (in comparison to the overall network). Moreover, we also consider the *share of bandwidth* that is retained. In order to decide when carrying out the flag day would be appropriate, both metrics need to be taken into account. Note that they stand for fundamentally different goals Tor tries to achieve. While the available bandwidth can be used as a rough metrics for Tor's performance, the number of available relays is crucial for the diversity of the network, directly impacting the level of anonymity Tor can offer. We obtain the necessary data by evaluating historical directory data from Tor Metrics—relays do not only publish their software version, but also the bandwidth that they are able to contribute to the Tor network.

Figure 3 shows the results of evaluating the historical data for the introduction of Tor version 0.2.5. Note that the share of nodes that are retained in a flag day is equal to the share of nodes that have upgraded at least to the target version.

Recall that this version was deliberately chosen as a worst case scenario. Consequently, our results show that the upgrade progress is slow over the network. After one year, still less than 10 percent of the relays have upgraded. On the other hand, these account for an disproportionally high fraction of network bandwidth that is available already. We deduce from this observation that high-bandwidth relays are more likely to be upgraded more quickly due to a higher level of admin activity or consciousness for the relevance of regular updates. Moreover, from the data, we observe that this is particularly the case for exit relays (not shown in the figure).

Despite the initially slow upgrade behavior, a clear rise can be observed approximately one and a half years after version 0.2.5 was initially published. Afterwards, half of the relays have upgraded, providing about 70 percent of the network bandwidth. The upgrade process continues slowly, the 90 percent landmark of upgraded relays is reached four years

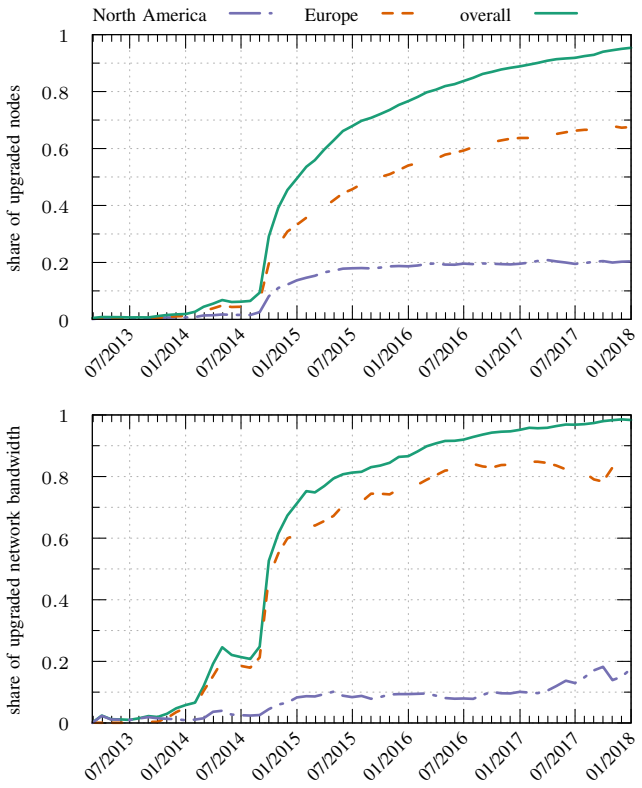

Fig. 3. Share of Tor network resources that would remain part of the network after a flag day transition, plotted over time. We assume versions 0.2.4 and below to be legacy.

after the initial release (March 2017).

Also note that there is a temporary decline in upgrade progress around April 2014. This effect is closely related to the *Heartbleed* vulnerability [24] that affected OpenSSL and thus also the Tor software. After not having applied appropriate fixes, the Tor directory authorities decided to exclude relays from the network that were still vulnerable [25]. Due to the fact that several old versions at that time were not affected by Heartbleed because they were relying on too old OpenSSL versions, this resulted in a temporary shift towards software of older versions being active in the network.

When evaluating the retained network resources, we also considered their geographical distribution to assess network diversity. Therefore, Figure 3 also distinguishes between resources located in Europe and North America, respectively. Together, these two regions account for the overwhelming majority of resources within the Tor network. We observe that the flag day does not introduce any significant additional imbalance to the network. The results reflect the existing bias towards European relay operators, which is especially prominent in the case of network bandwidth. However, the ratio between European and North American resources stays nearly constant throughout the time span we considered. This hints at a relatively similar upgrade behavior in these regions.

While these statistics reveal that a flag day strategy can

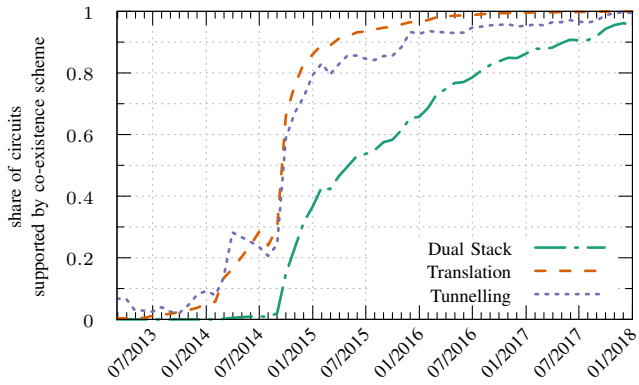

Fig. 4. Share of circuits that support the respective co-existence scheme. Results are based on the introduction of version 0.2.5.

potentially require a considerable amount of time until a protocol change has been distributed sufficiently for switching over, they also show that, for far-reaching changes to the network, a flag day may constitute a conceivable strategy.

This is especially true when taking into account the novel versioning scheme that was introduced for Tor with the version we considered. The novel approach of periodically releasing LTS versions of overlapping support periods is expected to lower the time it takes for new versions to be adopted. At the same time, it leads to a more diverse network, in which several different versions are active. This, in turn, makes it more difficult to effectively apply a co-existence scheme. In this regard, a flag day transition can serve as a serious alternative.

However, it remains unclear to what degree a flag day at any time would affect end users, running the Tor client software. Due to the fact that this factor cannot currently be assessed as necessary, we conclude that carrying out flag day changes in the Tor network currently remains infeasible.

C. Co-existence

We now evaluate to what extent deployment schemes based on co-existence enable the use of the novel protocol extension during the transition phase, at which only parts of the network have upgraded and circuits may thus comprise both, legacy and upgraded relays.

Figure 4 presents the results we obtained from running our TorPS simulations. Based on how often each of the circuit configurations from Table II occurred in the simulation, it depicts how well each of the co-existence schemes under consideration is suitable to enable the partial activation of the protocol change during the transition phase.

The first thing that can be noted is that both, translation and tunneling, are considerably more effective than dual stack. As the dual stack approach can only be used for circuits in which all relays have already upgraded, this was is in line with our expectations. In contrast, by applying a tunneling or translation deployment scheme, an increasing fraction of established circuits in the network can apply the protocol change at least to a part of the circuit earlier in the deployment

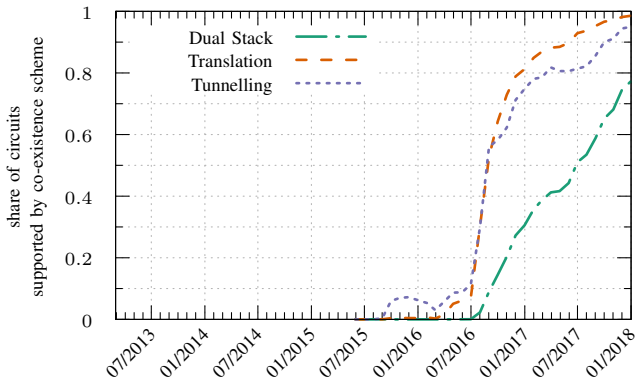

Fig. 5. Share of circuits that support the respective co-existence scheme. Results are based on the introduction of version 0.2.8.

process. For example, one and a half years after the new version was released, already 60 percent of the circuits could have, in parts, utilized it when applying a translation or tunneling scheme. At the same time, dual stack would only enable it for approximately 10 percent. The degree to which this makes sense and is useful in the spirit of the protocol change, of course, depends on the change itself. Moreover, as we have noted before, the precise circuit configurations that are suitable for each deployment strategy may also vary.

Nevertheless, we see that applying a deployment scheme that does not require the complete circuit to be upgraded can result in a clear improvement of applicability of the protocol extension during the transition phase. While this does not imply that the deployment of changes to the Tor network can now be performed by mechanically applying a pre-defined rule set, it validates the approach of rolling out protocol changes to the Tor network incrementally by relying on general strategies that help to retain backwards compatibility with legacy nodes where possible.

To further validate this result, we also simulated the introduction of a more recent version (Tor version 0.2.8), shown in Figure 5. Our key observations remain the same. In addition, the results show that the new update policy in fact leads to lower deployment times, even when ignoring the effects of Heartbleed on version 0.2.5.

VI. Lessons Learned and Conclusion

In this work, we explored the problem of deploying changes to the Tor network. We find that, through its directory service and the extensible cell structure, Tor can in principle be extended to feature novel functionalities. Evolving the protocol at its core networking routines is a particularly challenging task, though. When altering its behavior at the link and transport layer special care must be taken to maintain maximum possible compatibility with legacy nodes. Otherwise, the consequences on anonymity may be severe. We therefore developed various deployment strategies and analyzed their applicability and impact on the Tor network. To this end, we provide a set of potential tools to ease the deployment of changes and validate their general applicability using a simulation study.

As a first strategy, we identified that a flag day transition could be used to deploy virtually any system change in a coordinated way. While being a very versatile and powerful strategy, we also see various risks and weaknesses, including network splits, Sybil attacks, and abandoning clients. This brings us to the conclusion that a flag day should currently not be considered as a deployment strategy for the Tor network.

As the second class of strategies, we came up with co-existence schemes, which allow a new protocol version to co-exist with the legacy system. In this regard, we introduce the notions of dual stack, translation and tunneling. We generally observe the transitional phase to be rather long in Tor. With our deployment strategies, though, a significant number of users might be able to benefit already from upgrades during that period.

However, our research also suggests that, with currently available techniques, one cannot always realize incremental deployment without partly sacrificing some of Tor's anonymity promises during the transitional phase. In essence, this is due to the risk of unintentionally making circuits distinguishable.

With our work, we also hope to raise awareness of the issue among the research community, motivating to integrate suitable deployment strategies directly into their Tor research.

Acknowledgment

Christoph Döpmann was supported by HU Berlin within the Excellence Initiative of the states and the federal government.

References

[1] R. Dingledine, N. Mathewson, and P. Syverson, "Tor: The second-generation onion router," in *USENIX Security '04: Proceedings of the 13th USENIX Security Symposium*, 2004.

[2] N. Mathewson, "Git merge of ntor handshake protocol into the Tor code base," Jan. 2013. [Online]. Available: https://gitweb.torproject.org/tor.git/commit/?id=b1bdecd703879ca09bf63bf1453a70c4b80ac96d

[3] M. AlSabah and I. Goldberg, "PCTCP: per-circuit TCP-over-IPsec transport for anonymous communication overlay networks," in *CCS '13: Proceedings of the 20th ACM Conference on Computer and Communications Security*, 2013.

[4] M. AlSabah, K. Bauer, I. Goldberg, D. Grunwald, D. McCoy, S. Savage, and G. Voelker, "DefenestraTor: Throwing out windows in Tor," in *PETS '11: Proceedings of the 11th Privacy Enhancing Technologies Symposium*, 2011.

[5] J. Reardon and I. Goldberg, "Improving Tor using a TCP-over-DTLS tunnel," in *USENIX Security '09: Proceedings of the 18th USENIX Security Symposium*, 2009.

[6] F. Tschorsch and B. Scheuermann, "Mind the gap: Towards a backpressure-based transport protocol for the Tor network," in *NSDI '16: Proceedings of the 13th USENIX Symposium on Networked Systems Design and Implementation*, 2016.

[7] A. Carzaniga, A. Fuggetta, R. S. Hall, D. Heimbigner, A. van der Hoek, and A. L. Wolf, "A characterization framework for software deployment technologies," Colorado State University, Tech. Rep., 1998.

[8] A. Dearie, "Software deployment, past, present and future," in *FOSE '07: Proceedings of the 7th Workshop on the Future of Software Engineering*, 2007.

[9] C. Pahl, "Containerization and the PaaS cloud," *IEEE Cloud Computing*, vol. 2, no. 3, 2015.

[10] D. Thaler, "Planning for protocol adoption and subsequent transitions," Internet Requests for Comments, RFC Editor, RFC 8170, 2017.

[11] E. Cordeiro, R. Carnier, and W. L. Zucchi, "Comparison Between IPv4 to IPv6 Transition Techniques," *arXiv e-print arXiv:1612.00309*, 2016. [Online]. Available: http://arxiv.org/abs/1612.00309

[12] M. AlSabah and I. Goldberg, "Performance and security improvements for Tor: A survey," *ACM Computing Surveys (CSUR)*, vol. 49, no. 2, 2016.

[13] R. Dingledine, N. Mathewson, and P. Syverson, "Challenges in deploying low-latency anonymity," Tech. Rep., 2005.

[14] R. Jansen and M. Traudt, "Tor's been KIST: A case study of transitioning Tor research to practice," *arXiv e-print arXiv:1709.01044*, 2017. [Online]. Available: http://arxiv.org/abs/1709.01044

[15] D. M. Goldschlag, M. G. Reed, and P. F. Syverson, "Hiding Routing Information," in *IHW '01: Proceedings of the 1st International Workshop on Information Hiding*.

[16] D. L. Chaum, "Untraceable electronic mail, return addresses, and digital pseudonyms," *Communications of the ACM*, vol. 24, no. 2, 1981.

[17] R. Dingledine and N. Mathewson, "Tor protocol specification." [Online]. Available: https://gitweb.torproject.org/torspec.git/tree/tor-spec.txt

[18] R. Dingledine, "Tor 0.0.6 is out," May 2004. [Online]. Available: https://lists.torproject.org/pipermail/tor-announce/2004-May/000063.html

[19] The Tor Project, "Tor proposal process," Jan. 2007. [Online]. Available: https://gitweb.torproject.org/torspec.git/tree/proposals/001-process.txt

[20] N. Mathewson, T. Wilson-Brown, and A. Johnson, "Tor proposal 288: Privacy-preserving statistics with Privcount in Tor (Shamir version)," Dec. 2017. [Online]. Available: https://gitweb.torproject.org/torspec.git/tree/proposals/288-privcount-with-shamir.txt

[21] C. Viecco, "UDP-OR: A fair onion transport design," in *HotPETS '08: 1st Workshop on Hot Topics in Privacy Enhancing Technologies*, 2008.

[22] The Tor Project, "Tor Metrics Portal." [Online]. Available: https://metrics.torproject.org/

[23] A. Johnson, C. Wacek, R. Jansen, M. Sherr, and P. F. Syverson, "Users get routed: traffic correlation on tor by realistic adversaries," in *CCS '13: Proceedings of the 20th ACM Conference on Computer and Communications Security*, 2013.

[24] Z. Durumeric, J. Kasten, D. Adrian, J. A. Halderman, M. Bailey, F. Li, N. Weaver, J. Amann, J. Beekman, M. Payer, and V. Paxson, "The matter of heartbleed," in *IMC '14: Proceedings of the 14th ACM SIGCOMM Conference on Internet Measurement*, 2014.

[25] R. Dingledine, "Recommended reject lines for relays affected by heartbleed," Apr. 2014. [Online]. Available: https://lists.torproject.org/pipermail/tor-relays/2014-April/004362.html